第3章 電力，電磁波から暗号回路の内部動作を解析する

―― DES暗号回路を実測し，差分電力解析を行う

高橋芳夫，佐藤 証

ここでは，実際に暗号LSIからデータを収集し，どのようにして秘密かぎが抽出されるかを解説する．筆者らは，FPGAにDES(Data Encryption Standard)暗号回路を実装し，差分電力解析(DPA)を行った．その結果，数千サンプルのデータから，秘密かぎ全56ビットを取り出せることがわかった．なお，今回の実験は使用したFPGA固有の性質によるものではなく，LSIが動作する際の消費電力の変動や電磁波の放射など，一般的な現象を利用している．（編集部）

筆者らは，実際にDES暗号回路を動作させて消費電力と電磁波の観察を行い，LSIの漏えい情報を解析して秘密かぎを取り出す実験を行いました．ここでは，電力や電磁波の測定がどのように行われるのか，また測定波形からどのようなことが分析できるのかを解説します．

サイドチャネル攻撃(正規の入出力経路以外からの情報を利用した攻撃)の評価用として，耐タンパ性(攻撃に対して，秘密情報の守秘や機能の改変を困難にする性質)に関する標準化のための調査・研究を行っているINSTAC(Information Technology Reseach and Standardization Center)などが専用評価ボードを開発しています．こうした評価ボードには，高性能な電源回路やコンデンサ(OS-CONやPOSCAPなど)を使ってノイズを抑えたり，電力(電流)測定用にシャント抵抗を実装して精密に測定できるようにくふうされたものがありますが，非常に高価なうえに入手も困難です．

DES程度の小さな暗号回路で簡単な実験を行うには，このような専用ボードよりも，シンプルな構成のFPGAボードが適しています．そこで，今回の実験では本誌2005年1月号付属のFPGA基板を利用しました．なお，この実験はFPGA(XC3S50)に固有の機能や性質に依存したものでは

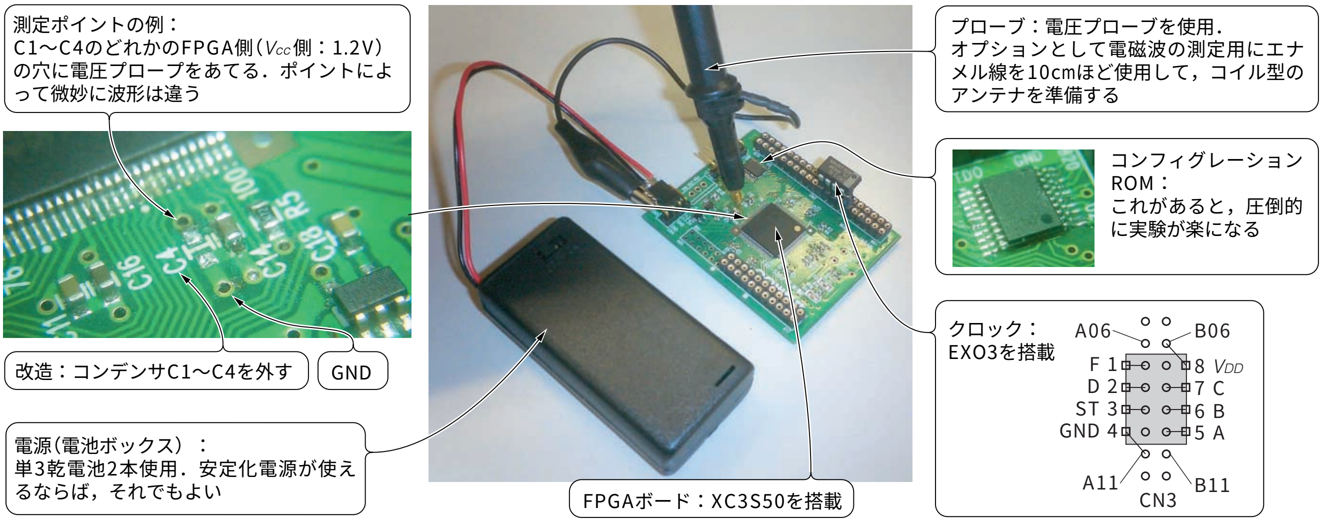


図1 実験環境
図に，今回の実験のために行ったFPGA基板の改造内容と測定環境をまとめる．

図2
DESのループ・アーキテクチャ
中間データを保持するための64ビットのレジスタ(Drg)，ラウンドかぎをラウンドごとに生成保持するための56ビットのレジスタ(Krg)，ラウンド制御用のレジスタ(Rrg)があり，クロックに同期して値が更新される．レジスタの値が変化すると，Drg [33：64]がE→XOR→S-Box→Pと処理され，Drg [1：32]とXORされた後，次のクロックでDrgに格納される．

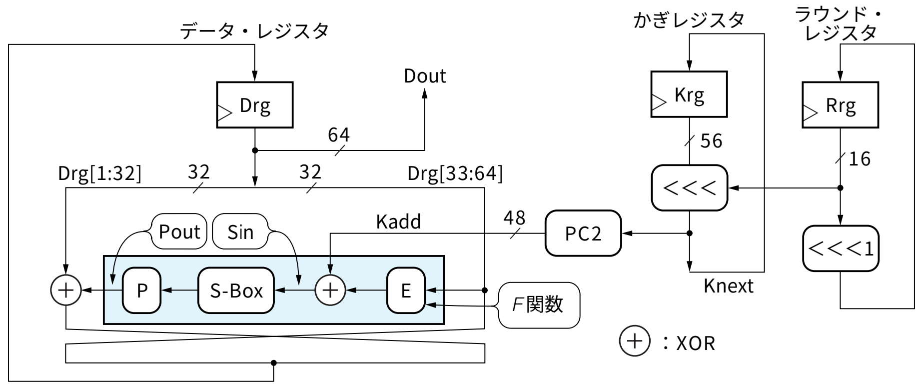


表1　主要部品

部　品	個 数	備　考
FPGA基板	1枚	本誌2005年1月号付属FPGA基板
コンフィグレーションROM	1個	使用を推奨
クロック発振器EXO3(16MHz)	1個	内部分周して4MHzを使用
丸ピンICソケット(1×20)	1本	CN2とCN3の必要部分に使用
L字型ヘッダピン(1×40)	1本	CN1とCN4(4ピン，6ピン)に使用
2ピン・ソケット	1個	電池ボックスの端子に使用
電池ボックス(単3，2本入り)	1個	—
乾電池(単3)	2本	—
エナメル線	少々	電磁波測定用(オプション)

ありません．LSIが動作する際の消費電力の変動や電磁波の放射という一般的な現象を利用したもので，ほかのデバイスでも同様に適用可能なものです．

1 サイドチャネル攻撃のための準備

以下に，今回の実験のために必要な準備を述べます．また，表1と図1に，実験で使用するおもな部品と改造内容，そして測定環境をまとめておきます．

● FPGA基板を準備する

まず，FPGA基板上にクロック発振器「EXO3(16MHz, DIP 8ピン型)」を載せます．ここでは，観測結果が良好であった内部分周の4MHzをクロックとして使います．EXO3の16MHzクロックをFPGA内で分周するほうが自由度は高いのですが，今回はDES以外の回路をできるだけ少なくするためにEXO3で内部分周します．もちろん，4 MHzのクロック・モジュールを実装してもかまいません．

クロック配線を短くするために，EXO3のV_{CC}とGNDピンを少し延ばしてコネクタCN3上に載せました(図1)．ただし，EXO3の仕様では電源電圧は5V±0.5Vなので，3.3 Vでの使用は実験目的以外では推奨できません．

電源は，FPGA基板の仕様では3.3Vとなっていますが，今回は単3の乾電池を2本直列につないで使用します．比較的軽い処理ということもあり，筆者の手元の3枚の基板では問題なく動作しています．スイッチング・レギュレータを使用したACアダプタは便利ですが，電源ラインにノイズが入るため，今回の実験には向きません．低ノイズの安定化電源が使えるのであれば，そちらを使用するのもよいでしょう．FPGA基板には3.3Vレギュレータ用のパターンがありますが，これは使用しませんでした．

そのほか，基板に用意されたパターンに従って，コンフィグレーションROM(U2)，電源供給用ヘッダ・ピン(CN4, 4ピン，3.3V側を使う)，コンフィグレーション用ヘッダ・ピン(CN1，6ピン)，トリガ出力用モニタ・ピン(CN2，1ピン)を取り付けます．

● FPGAにDES回路を実装する

DES暗号は，かぎを変形しながらラウンド処理を16回繰り返します(詳細は第2章を参照)．図2に示すように，今回の実装では1ラウンド分のデータパスを16回使用するループ・アーキテクチャと呼ばれる方式をとっています．また，サイドチャネル攻撃への対策や特別な高速化は行っていません．

DESのラウンド処理でLUT(look-up table)が割り当てられるのは，XOR(排他的論理和)とS-Boxとセレクタです．拡大転置Eや32ビット転置Pといったビットの転置は，単なる配線となります．XORは1ビット当たり1個のLUTを使用します．ただし，Pout(F関数の出力)とDrg [1：32]のXORは，データ・レジスタDrgのセレクタと合成されて1個のLUTとなります．また，S-Boxは6ビット入力，4ビット出力の置換テーブル八つで構成され，図3のよう

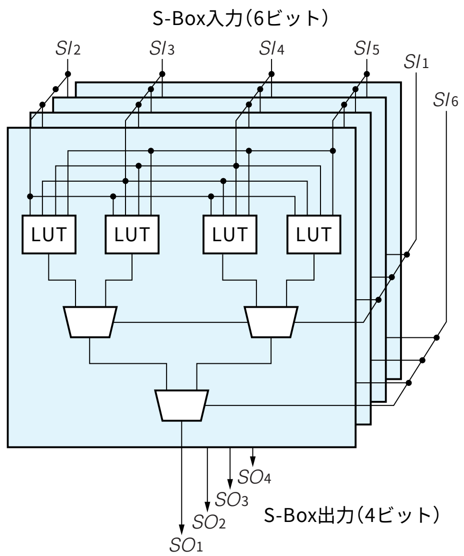


図3　FPGA上のS-Boxの構造

4ビット SI_2～SI_5は，それぞれ16個のLUTに入力される．16個のLUTから出力された16本の信号は SI_1で8本に，さらに SI_6で4本に絞られてS-Box出力となる．

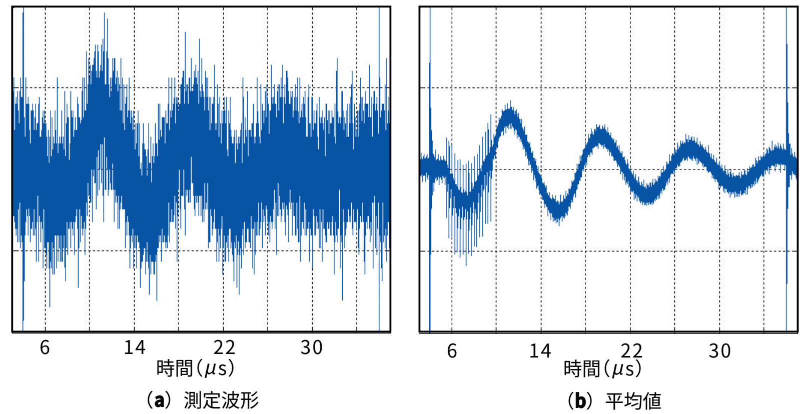


図5　バイパス・コンデンサがある状態での電圧波形

(a)は，普通の電圧プローブで測定した波形である．ノイズが大きく，どこでDESが動作しているのかは不明．(b)は，波形の平均を求めたものである．DESの動作を目視できる．

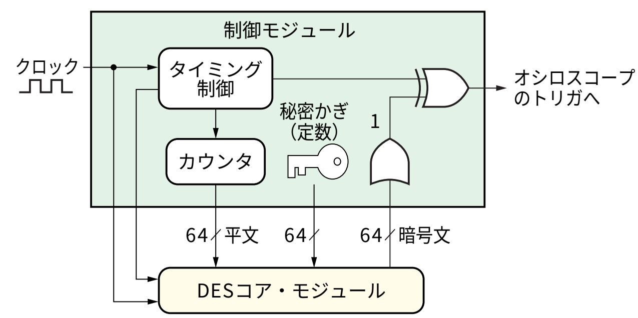


図4　DESの制御モジュールの構成

DESの入力となる平文は，制御モジュール内のカウンタで生成する．また，秘密かぎは固定値とする．DESの出力である暗号文64ビットをすべてORした後(通常‘1’になる)，DESの処理開始のタイミングを知らせるトリガとXORしてI/Oピンに出力する．これは，I/Oに影響しない回路が論理合成ツールで削除されるのを避けるためである．

に一つのテーブル当たり16個のLUTを使用して2段のマルチプレクサ(MUX)で4ビット出力を選択する構成となっています．

図4に，DESの制御モジュールを示します．電源ONとともに，カウンタによって内部生成した平文を順次暗号化する簡単なものです．なお，実験では秘密かぎは0x0123456789abcdefに固定しています．サイドチャネル攻撃では，オシロスコープなどで測定したさまざまな情報と暗号文(あるいは平文)から秘密かぎを推定します．暗号文64ビットのすべてをFPGA基板から出力するには外部回路が必要なのでそれは行わず，別途計算しておいた暗号文を使用します．

リスト1にDESコア・モジュールのVerilog HDLコードを示しておきます[注1]．

2 FPGA基板の漏えい情報を測定する

次に，サイドチャネル攻撃に有利なデータを取得するための基板の改造，および消費電力と電磁波の測定について解説します．

● 消費電力の測定の代わりに電圧変動を測る

消費電力の時系列変化の測定法として，GND側に数Ωの抵抗を直列に挿入し，その両端に発生する電位差をオシロスコープで観測する手法がよく紹介されています．ところが本FPGA基板では，FPGAの10本のGND端子がさまざまな場所でGNDプレーンに接続され，しかも一部のパターンはFPGAチップの下に入っています．これに対して，V_{CCINT}側は4本の端子に4個のバイパス・コンデンサが付加され，その先は1本のラインで1.2V用レギュレータにつながっています．そこで，この1.2V用レギュレータの出力電圧をオシロスコープで測定することにしました．

図5(a)に，**図1**の測定ポイントにおいて，オシロスコープの電圧プローブで測定した波形を示します．この図では，大きなリプルが発生しているのがわかるだけです．複

注1：リスト1のソース・コード(および，本特集第2章のリスト1のソース・コード)は，http://www.aoki.ecei.tohoku.ac.jp/crypto/からダウンロードできる．また，http://www.opencores.org/で公開されているフリーのDESコードでも実験可能なので，読者自身の手で試してみるのもよいだろう．

リスト1　DES コア・モジュール

S-Box の部分（SP モジュール）については，第2章のリスト1を参照のこと．

```
module DES(Din, Key, Dout, Drdy, Krdy, ENC, RSTn, EN, CLK, BSY, Dvld);
  input  [1:64] Din;  // Data input
  input  [1:64] Key;  // Key input
  output [1:64] Dout; // Data output
  input  Drdy;        // Data input ready
  input  Krdy;        // Key input ready
  input  ENC;         // 1 encryption, 0 decryption
  input  RSTn;        // Reset (Low active)
  input  EN;          // DES circuit enable
  input  CLK;         // System clock
  output BSY;         // Busy signal
  output Dvld;        // Data output valid

  reg  [1:64] Drg;    // Data register
  reg  [1:56] Krg;    // Key Register
  reg  [1:16] Rrg;    // Round Register
  reg  BSYrg;         // 0 WAIT, 1 ROUND (busy)
  reg  Dvldrg;

  wire [1:64] IP;
  wire [1:56] PC1, PC2, Knext;
  wire [1:48] Kadd, Sin;
  wire [1:32] Pout;
```

初期転置IPで64ビットの入力データをかき混ぜる

```
  assign IP   = {
Din[58], Din[50], Din[42], Din[34], Din[26], Din[18], Din[10], Din[2],
Din[60], Din[52], Din[44], Din[36], Din[28], Din[20], Din[12], Din[4],
Din[62], Din[54], Din[46], Din[38], Din[30], Din[22], Din[14], Din[6],
Din[64], Din[56], Din[48], Din[40], Din[32], Din[24], Din[16], Din[8],
Din[57], Din[49], Din[41], Din[33], Din[25], Din[17], Din[ 9], Din[1],
Din[59], Din[51], Din[43], Din[35], Din[27], Din[19], Din[11], Din[3],
Din[61], Din[53], Din[45], Din[37], Din[29], Din[21], Din[13], Din[5],
Din[63], Din[55], Din[47], Din[39], Din[31], Din[23], Din[15], Din[7]
};
```

データ出力時の最終転置IP^{-1}

```
  assign Dout = {
Drg[8], Drg[40], Drg[16], Drg[48], Drg[24], Drg[56], Drg[32], Drg[64],
Drg[7], Drg[39], Drg[15], Drg[47], Drg[23], Drg[55], Drg[31], Drg[63],
Drg[6], Drg[38], Drg[14], Drg[46], Drg[22], Drg[54], Drg[30], Drg[62],
Drg[5], Drg[37], Drg[13], Drg[45], Drg[21], Drg[53], Drg[29], Drg[61],
Drg[4], Drg[36], Drg[12], Drg[44], Drg[20], Drg[52], Drg[28], Drg[60],
Drg[3], Drg[35], Drg[11], Drg[43], Drg[19], Drg[51], Drg[27], Drg[59],
Drg[2], Drg[34], Drg[10], Drg[42], Drg[18], Drg[50], Drg[26], Drg[58],
Drg[1], Drg[33], Drg[ 9], Drg[41], Drg[17], Drg[49], Drg[25], Drg[57]
};
```

秘密かぎの初期転置PC1．64ビット入力のうち，パリティの8ビットはここで捨てられる

```
  assign PC1  = {
Key[57], Key[49], Key[41], Key[33], Key[25], Key[17], Key[ 9],
Key[ 1], Key[58], Key[50], Key[42], Key[34], Key[26], Key[18],
Key[10], Key[ 2], Key[59], Key[51], Key[43], Key[35], Key[27],
Key[19], Key[11], Key[ 3], Key[60], Key[52], Key[44], Key[36],
Key[63], Key[55], Key[47], Key[39], Key[31], Key[23], Key[15],
Key[ 7], Key[62], Key[54], Key[46], Key[38], Key[30], Key[22],
Key[14], Key[ 6], Key[61], Key[53], Key[45], Key[37], Key[29],
Key[21], Key[13], Key[ 5], Key[28], Key[20], Key[12], Key[ 4]
};
```

EC==1で暗号化．レジスタRrgが示すラウンドに応じて，かぎスケジュールのシフト量を変える

```
  assign Knext = (ENC == 1)?
     (((Rrg[1] | Rrg[2] | Rrg[9] | Rrg[16]) == 1)?
       {Krg[2:28], Krg[1], Krg[30:56], Krg[29]}:
       {Krg[3:28], Krg[1:2], Krg[31:56], Krg[29:30]}):
     (((Rrg[1] | Rrg[8] | Rrg[15] | Rrg[16]) == 1)?
       {Krg[28], Krg[1:27], Krg[56], Krg[29:55]}:
       {Krg[27:28], Krg[1:26], Krg[55:56], Krg[29:54]});

  assign PC2 = (ENC ==1)? Knext: Krg;
```

ラウンドかぎ出力時の56ビット→48ビット縮約転置PC2

```
  assign Kadd = {
PC2[14], PC2[17], PC2[11], PC2[24], PC2[ 1], PC2[ 5],
PC2[ 3], PC2[28], PC2[15], PC2[ 6], PC2[21], PC2[10],
PC2[23], PC2[19], PC2[12], PC2[ 4], PC2[26], PC2[ 8],
PC2[16], PC2[ 7], PC2[27], PC2[20], PC2[13], PC2[ 2],
PC2[41], PC2[52], PC2[31], PC2[37], PC2[47], PC2[55],
PC2[30], PC2[40], PC2[51], PC2[45], PC2[33], PC2[48],
PC2[44], PC2[49], PC2[39], PC2[56], PC2[34], PC2[53],
PC2[46], PC2[42], PC2[50], PC2[36], PC2[29], PC2[32]
};
```

データ・レジスタの半分Drg[64：33]を拡大転置Eで48ビットに拡大した後にラウンドかぎとXOR

```
  assign Sin = {
Drg[64], Drg[33:37], Drg[36:41], Drg[40:45],
Drg[44:49],Drg[48:53], Drg[52:57], Drg[56:61],
Drg[60:64], Drg[33]
} ^ Kadd;
```

ラウンドかぎとのXOR後，S-Box変換と転置Pを実行．PoutはF関数出力となる

```
  SP SP (Sin, Pout);

  assign BSY = BSYrg;
  assign Dvld = Dvldrg;

  always @(posedge CLK) begin
    if (RSTn == 0) begin
      Rrg    <= 16'b1000000000000000;
      BSYrg  <= 0;
      Dvldrg <= 0;
    end
```

16ラウンドの処理は16ビットのシフト・レジスタで制御

```
    else if (EN == 1) begin
      if (BSYrg == 0) begin  // Idle
        if (Krdy == 1)
          Krg <= PC1;
        if (Drdy == 1) begin
          BSYrg  <= 1;
          Dvldrg <= 0;
          Drg    <= IP;
        end
      end
```

*F*関数出力をデータ・レジスタの半分Drg[1：32]にXORし，左右の32ビットを入れ替えてDrgにラッチ

```
      else begin  // Round
        Drg <= {Drg[33:64], Drg[1:32] ^ Pout};
        Krg <= Knext;
        Rrg <= {Rrg[16], Rrg[1:15]};
        if (Rrg[16] == 1) begin
          BSYrg  <= 0;
          Dvldrg <= 1;
        end
      end
    end
  end
endmodule
```

シフトによってRrg[1]=1→Rrg[16]=1となれば，16ラウンドが終了

数の波形を平均すると，**図5(b)**のようにDESのラウンド処理が波形として現れてきます．平均すると，DES動作時の波形が見えるので，原理的には本FPGA基板に差分電力解析(DPA：differential power analysis)を適用できます．しかし現実問題として，**図5**の状態では高性能なオシロスコープで精度の高い測定を行わないかぎり解析は難しいでしょう．

そこで，このリプルを抑え，かつ電圧変動の高周波成分がカットされるのを避けるため，基板上に実装されているバイパス・コンデンサC1～C4を取り外します．すると**図6**に示すように，波形を平均しなくてもDES動作が電圧変動としてわかるようになります．なお，この状態ではFPGAが正常動作する保証はありませんが，今回の実験では問題ありませんでした．

● 電磁波の測定にはループ・アンテナを用いる

電界プローブや磁界(あるいは電流)プローブと呼ばれるセンサを用いて，電磁波を電圧に変換してオシロスコープで測定する方法があります．そのための高精度な測定システムも販売されています．

しかし，これらを使用するまでもなく，**写真1**のような手巻きのループ・アンテナを使って，DES動作中の電磁波をきれいにとらえることができます(**図7**)．ループ・アンテナによる電磁波の検出方法は，EMC(electro-magnetic compatibility)/EMI(electro-magnetic interference)対策時の簡易な手段としてよく紹介されています．また，サイドチャネル攻撃の実験でもいくつか報告があります．

電磁波の測定では，電圧測定のようにバイパス・コンデンサを外す必要はありません．一方で，電磁波はFPGA基板上の位置によって受信できる波形が異なるため，DES動作を反映した波形を受信できるように微調整が必要です．アンテナが1mmずれても振幅が大きく変わるため，調整にはある程度の根気が要ります．

3 FPGA基板の電圧変動を観察する

電力解析攻撃を行う前に，まずFPGA基板の電圧変動のようすを観察し，測定がうまくいっているかどうかを調べてみましょう．

図8の上側の波形は，DES動作時の電圧変動を表します．のこぎりの歯のように見える部分がラウンド処理に対応し

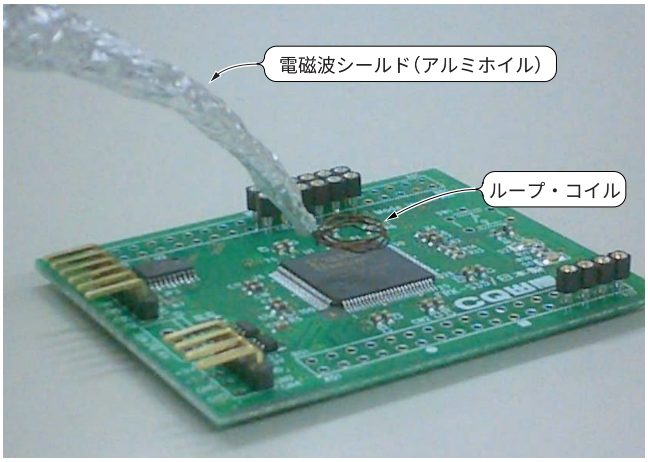


写真1　手巻きループ・アンテナによる電磁波プローブ
プローブをFPGA基板にかざしたようす．銀色の柄は周囲の電磁波をシールドするためのアルミホイル．

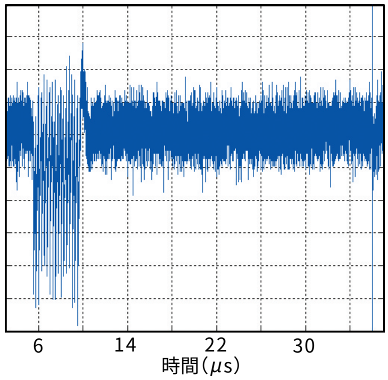


図6　バイパス・コンデンサを外した電圧波形
平均化しなくてもDES動作時の電圧変動がはっきりとわかる．

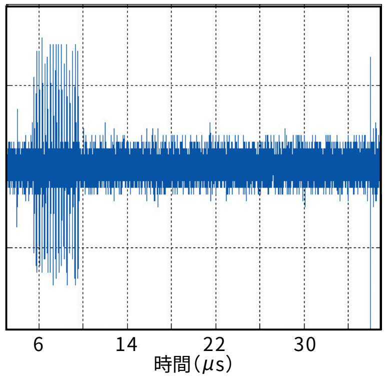


図7　電磁波形
電磁波プローブの場合，バイパス・コンデンサが付いていてもDESの動作を観察できる．

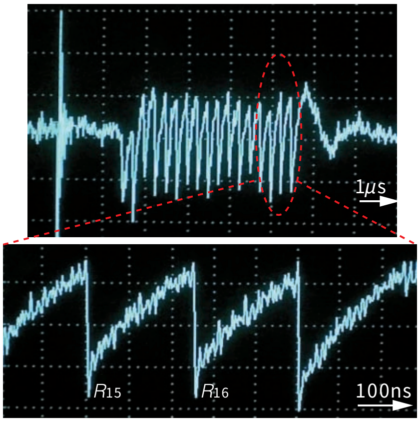


図8　DES動作時の電圧変動
図の上側がDES動作全体の電圧変動を，下側がラウンド処理の部分を拡大したものを表す．

ています．図8の下側は，その16ラウンド目の前後を拡大したものです．電圧が急激に低下している区間は，データ・レジスタDrgとかぎレジスタKrgが更新され，組み合わせ回路が動作している部分です．組み合わせ回路のスイッチングが終わると，キャパシタンスが徐々に充電されて電圧が回復します．クロックを速くすると，前回の電圧変化が回復する前に次の処理が始まるため，のこぎり波が斜め下にずれていきます．これを回避し，かつ計測時間が短くなるちょうど良いクロックとして，今回は4MHz（250 ns）を選択しました．

なお，図8の波形で山の数を数えると17個ありますが，最後の一つはデータ・レジスタDrgに最終結果をラッチしてDoutとして出力する際に，Drgの変化につられてラウンド処理部も動作するためです．

● 16ラウンド目のビット変化で電圧変動を確認する

CMOS回路の電力の大半はゲートのスイッチング時に消費されます．そこで，次のようなケースにおける平文を求め，この二つを入力としたときの電圧変動を調べました．

1）*F*関数の入力であるDrg [33：64] が16ラウンド目でまったく変化しない場合
2）16ラウンド目で全ビット変化する場合

図9から，**1）**の場合は電圧変動が少なく，**2）**のようにDrg [33：64] の変化が大きいときには電圧変動も大きくなることがわかります．この特徴を利用して，電圧波形を観察しながら変動が最小となる平文を探し出すことで，ラウンドかぎを求める能動的な攻撃が可能であるという報告もされています．これは単純電力解析（SPA ：simple power analysis）の一種ですが，決して「単純な攻撃」ではありません．平文をうまく選択するには暗号アルゴリズムを熟知し，デバイスの特徴も詳細に把握しているという高いスキルが要求されます．

これに対して，これから行うDPAでは，LSIが外部に出力する暗号文と動作波形だけを用いた受動的な攻撃が可能です．つまり，DPAアルゴリズムと呼ばれる一定の手順に従い，暗号文と測定データを統計処理するだけで秘密かぎを導出できるのです．

DPAでは特殊なパターンの中間データを生成する平文の選択は不要であり，入力は任意の値でかまいません．その場合，Drg [33：64] の各ビットはほぼ1/2の確率でランダムに変化するので，Drg [33：64] のビット・パターンは16±数ビットの範囲で変化します．それに応じて，図9の電圧もゆれ動きます．そこで，たくさんの波形を集めることで特定ビットの変化に依存した変動量を抽出しようというのがDPAです．

● 測定データの収集と差分グラフによる確認

DESの入力を0～4,095まで順番にインクリメントさせながら4,096個の測定データを収集し，パソコンに保存します．これを解析してかぎを導出するわけですが，その前にちょっとズルをして，内部状態（実験なのでFPGAにセットしたかぎを用いて計算できる）から測定がうまくいっているかどうかを簡単な統計処理で確認してみましょう．

第*n*ラウンドにおいて，Drg[33：64]の*i*ビット目が反転するかしないかによって測定データを2グループに分けて，そ

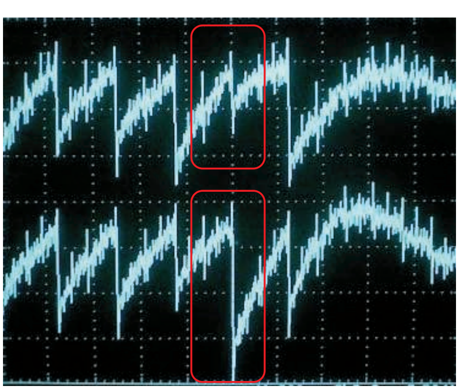


図9　入力パターンの違いによるDESの16ラウンド目の波形の比較
赤い線で囲んだ部分が16ラウンド目に対応する電圧変動．上側がレジスタDrg [33：64] が変化しない場合の波形．下側がDrg [33：64] が全ビット反転した場合の波形．

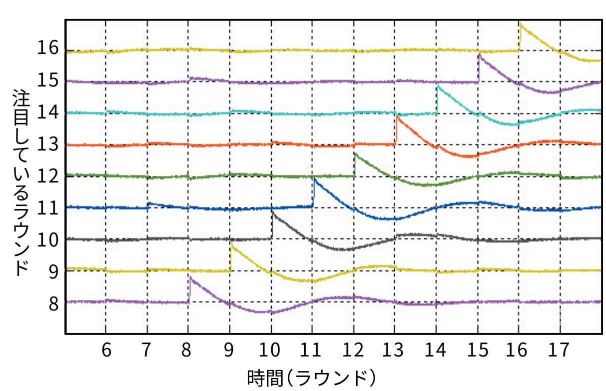


図10　第8～16ラウンドの差分グラフ
横軸はラウンド数，縦軸は差分グラフで注目したラウンド．第*n*ラウンドの差分グラフは，*n*ラウンドを実行しているときにだけ大きく変化する．

れぞれを平均した後に，その2グループの平均の差をとったものを**図10**に示します．第nラウンドにおいて大きく変動しているのは第nラウンドの差分グラフだけです．このことは，次に行うDPAにおいて推定したかぎの値が正しいときのみ差分が大きくなり，かぎの値が正しくないときには差分が0になることと対応します．したがって，この特徴が現れた測定データを用いたDPAは，うまくいきそうです．

4 DESの秘密かぎをDPAで導出する

いよいよDPAを行いますが，ここで攻撃者が満たすべき前提条件として次の2点を確認しておきましょう．

◉COLUMN◉ 標準的な選択関数を用いてDPAを行う

DPAは，対象とする暗号アルゴリズムの仕様に基づいて選択関数を構成できれば，実装の詳細は不明でも適用できる強力な攻撃です．そして，DESに対して次の選択関数が有効とされています．

$SF = Bit(SBox(X \oplus K_{16}),\ b)$ ……………………………………(A-1)

$SF = Bit(R_{16} \oplus P(SBox(X \oplus K_{16})),\ b)$

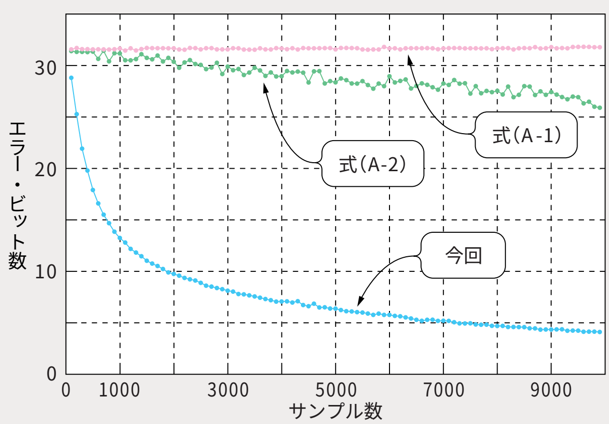
$= Bit(L_{15},\ b) = Bit(R_{14},\ b)$ ………………………………(A-2)

式(A-1)は，16ラウンド目のS-Box出力を利用してデータを分類します．また，式(A-2)は，15ラウンド目のF関数の入力を利用しています．例えば，DPAの提案者のKocher氏は式(A-2)を用いていますし，INSTAC(情報標準化研究センター)のZ80 CPUボードを用いたサイドチャネル攻撃に関する研究報告書では両方の式を使用しています．

ところが，今回の実験で両式を用いたところ，**表A-1**や**図A-1**に示すようにDPAはうまくいきませんでした．式(A-2)では正しい値がたまに出てくることもあるのですが，測定データを10,000サンプルに増やしてもほとんど改善が見られません．

この原因は，FPGAではS-BoxがANDやORといったプリミティブ・ゲートではなく，本文の**図3**のように4入力1出力のLUTとMUXで実装されているためと考えられます．ANDゲートやORゲートでは，入力または出力の1ビットが‘0’か‘1’かによって，ゲートのスイッチング確率に0.25の差があります．これが，DPAが成立する要因だと指摘されています．これに対して，八つのS-Boxを構成する，8×16個の4入力1出力のLUTにおいて，それぞれのスイッチング確率を調べてみたところ，**表A-2**のS1の例に示したように，すべての場合において確率は0.5でした．

また，MUXもランダムな入力をランダムに切り替えるだけなので，スイッチング確率は0.5となります．つまり，上記の選択関数では$SF=0$か$SF=1$かで，S-Box内部の信号のスイッチング確率に偏りはありません．そのため，今回用いたFPGA基板では，これらと異なる選択関数が必要となりました．



図A-1　選択関数によるDPA結果比較

表A-1　式(A-2)によるDPAの結果

S1	S2	S3	S4	S5	S6	S7	S8
39	41	16	3	22	32	38	26
1	38	46	56	52	7	61	5
50	32	7	61	46	47	27	50
19	3	7	3	35	42	43	56

サンプル数：10,000　エラー：26

S1を構成する
4入力テーブル部分

14	1	1	1	0
4	0	1	0	0
13	1	1	0	1
1	0	0	0	1
2	0	0	1	0
15	1	1	1	1
11	1	0	1	1
8	1	0	0	0
3	0	0	1	1
10	1	0	1	0
6	0	1	1	0
12	1	1	0	0
5	0	1	0	1
9	1	0	0	1
0	0	0	0	0
7	0	1	1	1

LSBに注目して
作表してみる

		前の状態(ランダムと仮定)															
		入力	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
今の状態(選択関数の対象)	入力	出力	0	0	1	1	0	1	1	0	1	0	0	0	1	1	0	1
	0	0			×	×		×	×		×				×	×		×
	1	0			×	×		×	×		×				×	×		×
	2	1	×	×			×			×		×	×	×			×	
	3	1	×	×			×			×		×	×	×			×	
	4	0			×	×		×	×		×				×	×		×
	5	1	×	×			×			×		×	×	×			×	
	6	1	×	×			×			×		×	×	×			×	
	7	0			×	×		×	×		×				×	×		×
	8	1	×	×			×			×		×	×	×			×	
	9	0			×	×		×	×		×				×	×		×
	A	0			×	×		×	×		×				×	×		×
	B	0			×	×		×	×		×				×	×		×
	C	1	×	×			×			×		×	×	×			×	
	D	1	×	×			×			×		×	×	×			×	
	E	0			×	×		×	×		×				×	×		×
	F	1	×	×			×			×		×	×	×			×	

▶表A-2
LUTにマッピングしたS-Boxのスイッチング

×印はスイッチングがある場合を示す．例えば，今の状態が「入力＝0」(いちばん上の行の状態)のとき，スイッチングがあるのは前の状態が「入力＝2，3，5，6，8，C，D，F」の8個で，ちょうど1/2である．ほかの状態でも同じである．

1）暗号装置内でDESが動作していることを知っている．DESのアルゴリズムは既知であるが，それ以外の装置内部の知識は持たない．

2）DES動作中の測定データと暗号文のペアを取得できる．ただし，秘密かぎや中間データなど内部データには直接アクセスできない．

この前提を満たせば，詳細な設計図や回路図などを知らなくても攻撃できてしまうという点が，DPAが強力であると言われる理由です．なお，今回の実験では次の条件も追加しておきます．

3）暗号装置のDESがFPGA上にループ・アーキテクチャで実装されていることを知っている．

これらの条件のもと，測定データと暗号文（話を簡単にするため，別途計算したもの）を用いてDPAを行います．

● DPAアルゴリズムを繰り返し実行してかぎを求める

DPAのアルゴリズムを**リスト2**に示します．

入力`m[ ]`は暗号文，`w[ ]`はそれに対応する測定データ，`N`は暗号文の測定データのサンプル数です．`b`は選択関数で使用するビット位置を示します．そして，`k`は48ビットのラウンドかぎのうち6ビットの推定値です．

6，7行目の`SF`は選択関数です．`k`のすべてのかぎ候補（0～63）に対して，`SF`が‘0’か‘1’かによって`t[0]`あるいは`t[1]`に測定データを加算します．そして，それぞれ加算した個数を`s[0]`と`s[1]`にカウントしておきます．したがって，9行目の`t[0]/s[0]`と`t[1]/s[1]`は`SF`＝0と`SF`＝1に対する測定データの平均であり，`v[ks]`がその差になります．その後，11行目で`v[ks]`の絶対値を最大値とする`ks`を，かぎビットの推定値として選びます．

DPA_ALGを1回実行すると，48ビットのラウンドかぎ中の6ビット（の推定値）が得られます．うまくいけば，注目するビット位置`b`を変えて8回実行するだけで，正しいラウンドかぎ48ビットが求まります．

リスト2 DPAアルゴリズム

```
1: Function DPA_ALG(m[], w[], N, b):k
2:   v[]=0;
3:   for ks=0..63 do begin
4:     t[]=0; s[]=0;
5:     for i=1..N do begin
6:       t[SF(m[i],ks,b)]+= w[i];
7:       s[SF(m[i],ks,b)]+=1
8:     end;
9:     v[ks] = t[1]/s[1] - t[0]/s[0];
10:   end;
11:   k = index of max(abs(v))
12: end;
```

● 最終ラウンドの値から選択関数が得られる

選択関数は，Drg［33：64］のある1ビットに注目して，16ラウンド目でそれが反転したならば$SF=1$，そうでなければ$SF=0$となるように定めます（通常の式と異なる理由は，p.121のコラム「標準的な選択関数を用いてDPAを行う」を参照）．Drg［33：64］の1ビットの反転の有無は，S-Box入力の1ビットの反転と相関があります．**図3**のS-Box（入力SI_1～SI_6，出力SO_1～SO_4）において，例えばビットSI_2に注目するとき，SI_3～SI_5がランダムに変化するならば，LUTのスイッチング確率はSI_2が反転するか否かに依存しているはずです．

なお，**図11**でF関数の出力とXORされるDrg［1：32］の$L_{14}\rightarrow L_{15}$の変化も電圧変動に影響します．しかし，これはタイミング・シミュレーションの結果から，F関数の出力変化よりもずっと早い時刻に起こることがわかっているので，選択関数には組み込んでいません．また，ラウンドかぎのK_{15}からK_{16}への変化はつねに一定なので，この部分に関する差分は0だと考えられます．

上記の議論のもと，選択関数を次のように$R_{14}\oplus R_{15}$のbビット目として定めました．

$$SF(m,\ k,\ b)=Bit(R_{14}\oplus R_{15},\ b) \quad \cdots\cdots\cdots(1)$$

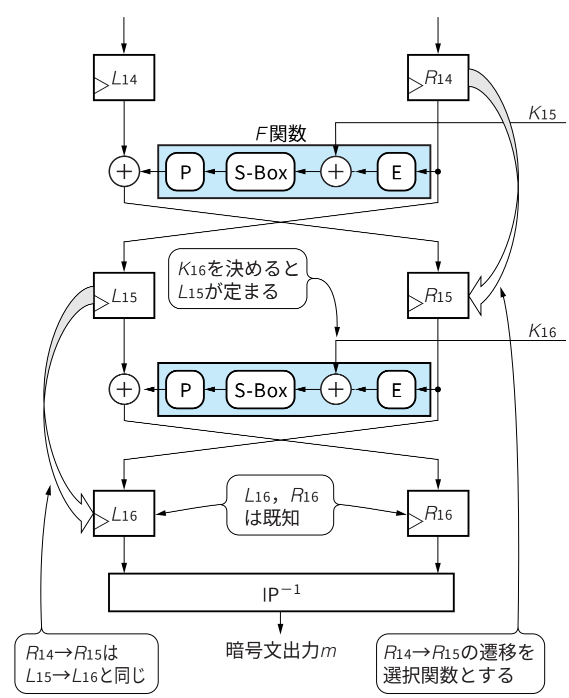


図11 DESの15～16ラウンドの処理

暗号文出力mからデータパスを逆にたどることで，内部状態のレジスタ遷移$R_{14}\rightarrow R_{15}$を選択関数として定めることができる．

ここで，mは暗号文(64ビット)，kはラウンドかぎ候補(6ビット，0～63)，bは注目するビット(1～32)です．DESの出力である暗号文mから最終ラウンドのL_{16}とR_{16}は計算できても，R_{14}とR_{15}は中間データなので，この選択関数は計算できないように思えます．しかし，式(2)のように変形することで，L_{16}とR_{16}，K_{16}から選択関数の値を求めることができます(**図11**)．

$$\begin{aligned} SF(m, k, b) &= Bit(R_{14} \oplus R_{15}, b) \\ &= Bit(L_{15} \oplus L_{16}, b) \quad \cdots (2) \\ &= Bit(R_{16} \oplus F(L_{16}, K_{16}) \oplus L_{16}, b) \end{aligned}$$

さらに，F関数をS-Box単位に分解すると，SFの計算には，R_{16}の1ビット，L_{16}の7ビット，K_{16}の6ビットがあればよいことがわかります．

● 3,000サンプルで56ビットかぎの探索が可能

S-Box1(S1)の出力4ビットが影響するパスの一つに注目して，**リスト2**で求めている差分値`v[ks]`(ks=0～63)の絶対値をグラフにしたものを**図12**に示します．ks＝50でピークが出ているので，S1に関係する6ビットのかぎは50であると推定できます．ほかのかぎビットも同じように求めることで，ラウンドかぎK_{16}の全ビットを推定できます．

表2は，八つのS-Box($S1$～$S8$)の各4ビット出力が影響するパスに注目して，サンプル数1,000個と3,000個でDPAを行った結果です．一つのS-Boxに対して6ビットのかぎ推定値が四つずつ求まります．**表2**では，いくつかの誤判定(灰色で示した部分)が起きていますが，多数決をとればすべて正しいかぎが得られます．また，サンプル数を増やすほど判定の精度は上がることがわかります．これで6ビット×8＝48ビットのラウンドかぎが判明しました．

次に，秘密かぎ(56ビット)の残り8ビットを総当たり(2^8＝256パターン)で調べます．この8ビットが求まらない場合は，下記のような方法をとります．

- DPAで誤判定を抑えるため，サンプル数を増やす
- 誤りが多かったS-Boxに対するかぎ6ビットを含めて，未定のビットを総当たりで探索する
- 補助的に別の選択関数を試してみる

なお，総当たりの探索では平文と暗号文のペアが少なくとも一つ必要ですが，平文が得られない場合，15ラウンド目の波形を用いたDPAで残りの8ビットを求めることもできます．

以上のように，本実験では，1,000～3,000サンプル程度の測定データによって，DESの秘密かぎ56ビットをすべて求めることができました．また，**図13**に示すように，電磁波の場合も同じ手順でほぼ同じ精度(エラー・レート)で秘密かぎを推定することができ，多数決ですべてのビット

表2 かぎの推定結果

S1	S2	S3	S4	S5	S6	S7	S8
63	35	7	1	48	7	0	13
50	35	52	3	46	7	0	39
50	61	52	3	39	43	0	50
53	34	6	3	46	7	0	50

(**a**) サンプル数：1,000　エラー：12

S1	S2	S3	S4	S5	S6	S7	S8
21	35	52	23	18	7	0	50
50	35	20	3	46	7	0	52
50	59	52	3	46	7	0	50
50	56	52	3	46	7	0	50

(**b**) サンプル数：3,000　エラー：7

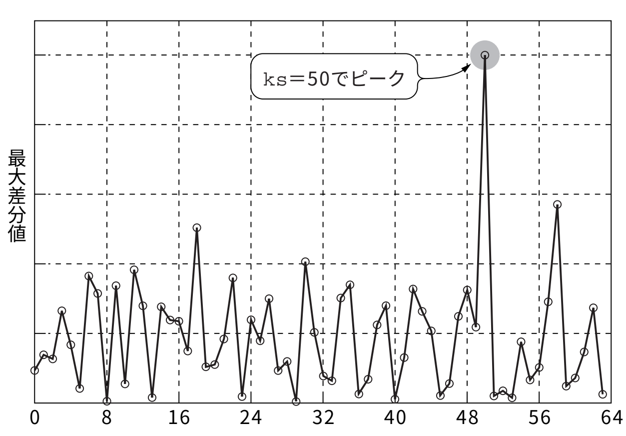


図12　かぎ候補ごとの最大差分値(サンプル数3,000)

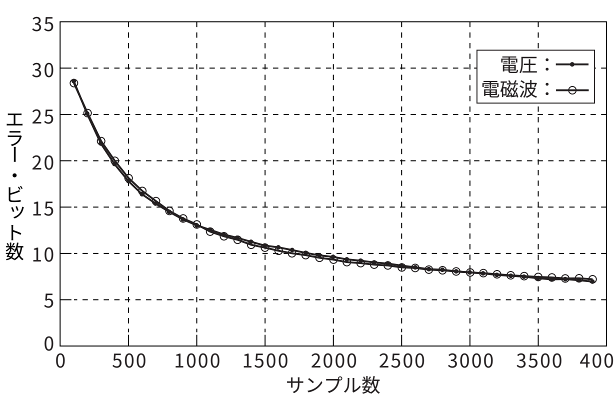


図13　電圧と電磁波のDPA結果

測定データのサンプル数を横軸に，エラーの数を縦軸にプロットした．電圧と電磁波でDPAの精度はまったく同じであることがわかる．

3

が正しく求まりました．

5 暗号回路設計にトライ！

本実験によって，LSIは電力消費や電磁波放射など，本来のデータ入出力チャネルではないサイドチャネルを介して，内部の動作状態を外に伝えていることがよくわかったと思います．1回の測定ではノイズに隠れてしまうわずかな情報でさえ，測定データを集めて統計処理することでかぎの情報を暴き出してしまいます．サイドチャネル攻撃は，同じかぎを繰り返し使用する暗号回路にとってまさに脅威です．しかし，この分野の研究がここ数年活発に行われ，攻撃法だけでなく対策法も数多く提案されています．そして，暗号処理を行う高機能な暗号装置などには，すでにいくつかの対策法が組み込まれています．

● 安全な暗号製品の実現に向けて

今回の実験で使用したFPGA基板は，電磁波をシールドするような筐体に入っていません．また，バイパス・コンデンサの除去や動作クロックの調節など，攻撃者にとって非常に有利な条件で動作波形の測定を行っています．したがって，今回の実験結果は，実際の暗号製品にそのまま適用できるとはかぎりません．また，従来の解析に使用されていた選択関数ではDPAが失敗したように，サイドチャネル攻撃がうまくいくかどうかは，LSIのアーキテクチャの違いにも大きく依存します[注2]．

サイドチャネル攻撃について実際のセキュリティ製品を解析するのは問題が多いため，アルゴリズムによる対策法は数多く提案されていますが，回路実装や測定法のノウハウはまだあまり公開されていません．また製品レベルでは，実装した対策法やアーキテクチャを公開しなければ攻撃者に与える情報が少ないので，今のところそのほうが安全と言えるかもしれません．

同じように，暗号アルゴリズムも非公開にしたほうが安全と考える人も少なくありません．しかし，非公開の暗号アルゴリズムは専門家による十分な検証が行われていないため，それが外部に漏れて，あっという間に破られてしまうといったケースも珍しくありません．そこで，AES（Advanced Encryption Standard）をはじめとする多くの標準アルゴリズムは，公開の場で解析評価が行われています．

サイドチャネル攻撃への対策法も，暗号アルゴリズムの安全性の検証と同じように，第三者による適切な評価が不可欠です．そこでISO/IEC 15408の評価機関では，そのためのノウハウの共有を進めています．また，米国のNIST（National Institute of Standards and Technology），そして日本のCRYPTREC（Cryptographic Research and Evaluation Committee）やINSTACも，サイドチャネル攻撃の研究・調査および暗号モジュールの実装の安全性に関する評価基準の策定に積極的に取り組んでいます．

暗号の研究は，これまで数学者（あるいは暗号学者）を中心に行われてきました．しかし，さまざまな家電製品に暗号が組み込まれ，またサイドチャネル攻撃が登場したことで，回路実装面からの研究が重視されてきています．ハードウェア・エンジニアのみなさんも暗号の世界に一歩足を踏み出してみてはいかがでしょうか．

注2：本誌2003年10月号付録のFPGA基板（米国Altera社のEPC1C3搭載）をお持ちの方は，そちらでも同じような実験を行ってみるとおもしろいだろう．アーキテクチャの違いが動作波形や解析結果にどう影響するかなど，きっと興味深い結果が得られるに違いない．

参考・引用*文献

(1) P. Kocher, J. Jaffe, B. Jun；Differential Power Analysis, CRYPTO'99, LNCS 1666, pp.388-397, Springer-Verlag, 1999.
(2) 森岡澄夫；HDLによる高性能ディジタル回路設計，p.102，CQ出版社，2002年.
(3) K. Gandolfi, C. Mourtel, F. Olivier；Electromagnetic Analysis：Concrete Results, CHES2001, LNCS 2162, pp.251-261, Springer-Verlag, 2001.
(4) 総力特集 第2章 本誌付属Spartan-3基板の概要，Design Wave Magazine，2005年1月号，pp.29-34.
(5) 酒井，佐伯，鈴木，佐藤；現実の脅威 サイドチャネル解析，日経エレクトロニクス，2005.7.18号〜2005.8.15号.

たかはし・よしお
(株)NTTデータ 技術開発本部
さとう・あかし
日本アイ・ビー・エム(株) 東京基礎研究所

<筆者プロフィール>

高橋芳夫． 大学卒業後，NTTデータ（当時，NTTデータ通信）に入社．製品開発と評価に従事する．CRYPTREC暗号モジュール委員となったのを契機にサイドチャネル攻撃に興味を持ち，現在勉強中．

佐藤証． 論理式からLSIのマスク・パターンを直接描く特殊能力を持つ私ですが，今回はその回路の電磁波をイメージできるステージへステップ・アップしました．一動作波形から暗号かぎをピタリと言い当てるレベルまで，あと一歩です．